

prooftrees

Version v0.5 (SVN Rev: 5576)

Clea F. Rees*

2016/12/06

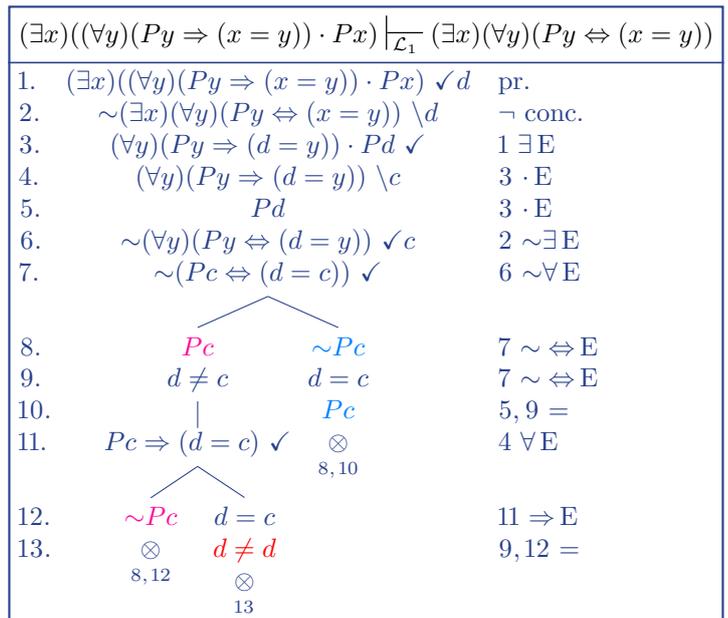
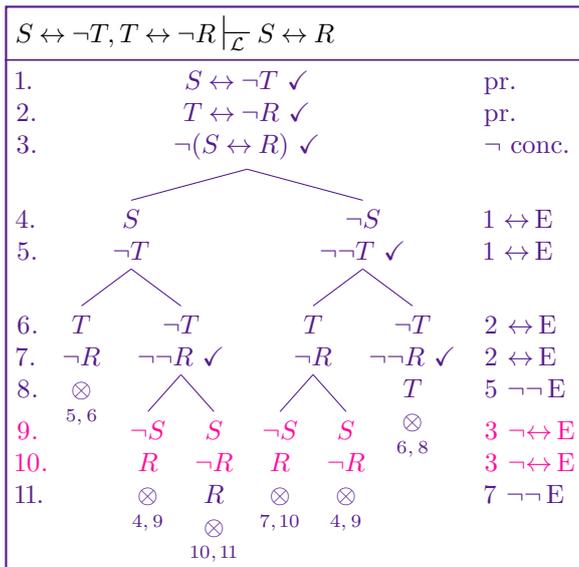
Abstract

prooftrees is a $\text{\LaTeX} 2\epsilon$ package, based on forest, designed to support the typesetting of proof trees in styles sometimes used in teaching introductory logic courses, especially those aimed at students without a strong background in mathematics. One textbook which uses proofs of this kind is Hodges (1977, 1991).

Note that this package requires version 2.1 (2016/12/04) of forest (Živanović 2016). It will not work with versions prior to 2.1.

I would like to thank Živanović both for developing forest and for considerable patience in answering my questions, addressing my confusions and correcting my mistakes. The many remaining errors are, of course, entirely my own. This package's deficiencies would be considerably greater and more numerous were it not for his assistance.

*reesc21 <at> cardiff <dot> ac <dot> uk



Contents

1	Raison d'être	2
2	Assumptions & Limitations	5
3	Typesetting a Proof Tree	5
4	Loading the Package	14
5	Invocation	14
6	Proof Tree Anatomy	14
7	Options	15
7.1	Global Options	15
7.2	Local Options	20
8	Macros	23
9	Version History	23

1 Raison d'être

Suppose that we wish to typeset a typical proof tree demonstrating the following entailment

$$\{P \vee (Q \vee \neg R), P \rightarrow \neg R, Q \rightarrow \neg R\} \vdash \neg R$$

We start by typesetting the tree using `forest`'s default settings (box 1) and find our solution has several advantages: the proof is specified concisely and the code reflects the structure of the tree. It is relatively

1
forest: default settings

```

\begin{forest}
  [$P \vee (Q \vee \neg R)$
    [$P \wedge \neg R$
      [$Q \wedge \neg R$
        [$\neg \neg R$
          [$P$
            [$\neg P$
            [$\neg R$
          ]
          [$Q \vee \neg R$
            [$Q$
              [$\neg Q$
              [$\neg R$
            ]
          ]
        ]
      ]
    ]
  ]
]
]
]
\end{forest}

```

```

graph TD
    A["P ∨ (Q ∨ ¬R)"] --- B["P → ¬R"]
    B --- C["Q → ¬R"]
    C --- D["¬¬R"]
    D --- E["P"]
    D --- F["Q ∨ ¬R"]
    E --- G["¬P"]
    E --- H["¬R"]
    F --- I["Q"]
    F --- J["¬R"]
    I --- K["¬Q"]
    I --- L["¬R"]

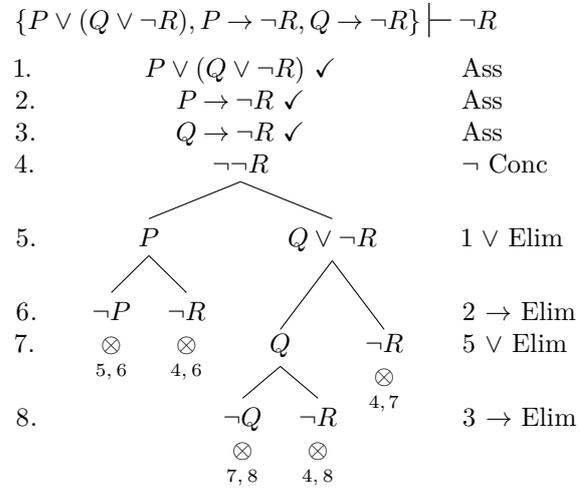
```

2 prooftrees: default settings

```

\begin{prooftree}
{
  to prove={\{P \vee (Q \vee \lnot R), P \lif
\lnot R, Q \lif \lnot R\} \sststile{}{\} \lnot
R}
}
[P \vee (Q \vee \lnot R), just=Ass, checked
[P \lif \lnot R, just=Ass, checked
[Q \lif \lnot R, just=Ass, checked,
name=last premise
[\lnot\lnot R, just={\lnot$ Conc},
name=not conc
[P, just={\vee$ Elim:!uuuu}
[\lnot P, close={:!u,!c}]
[\lnot R, close={:not conc,!c},
just={\lif$ Elim:!uuuu}]]
[Q \vee \lnot R
[Q, move by=1
[\lnot Q, close={:!u,!c}]
[\lnot R, close={:not conc,!c},
just={\lif$ Elim:last premise}]]
[\lnot R, close={:not conc,!c},
move by=1, just={\vee$ Elim:!u}]]]]]]
\end{prooftree}

```



straightforward to specify a proof using forest's bracket notation, and the spacing of nodes and branches is automatically calculated.

Despite this, the results are not quite what we might have hoped for in a proof tree. The assumptions should certainly be grouped more closely together and no edges (lines) should be drawn between them because these are not steps in the proof — they do not represent inferences. Preferably, edges should start from a common point in the case of branching inferences, rather than there being a gap.

Moreover, proof trees are often compacted so that *non-branching* inferences are grouped together, like assumptions, without explicitly drawn edges. Although explicit edges to represent non-branching inferences are useful when introducing students to proof trees, more complex proofs grow unwieldy and the more compact presentation becomes essential.

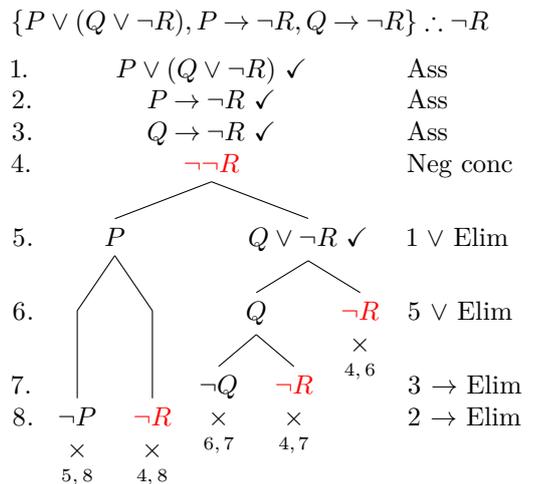
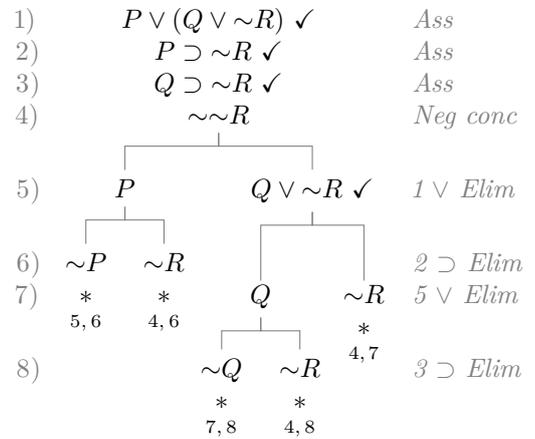
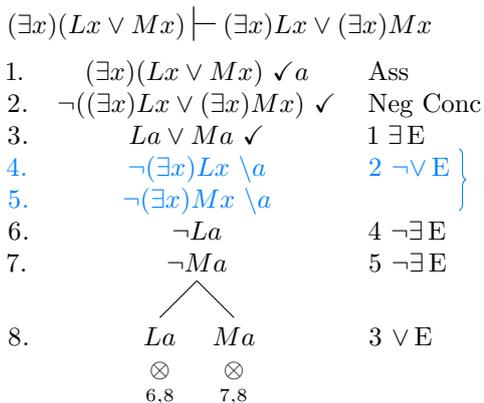
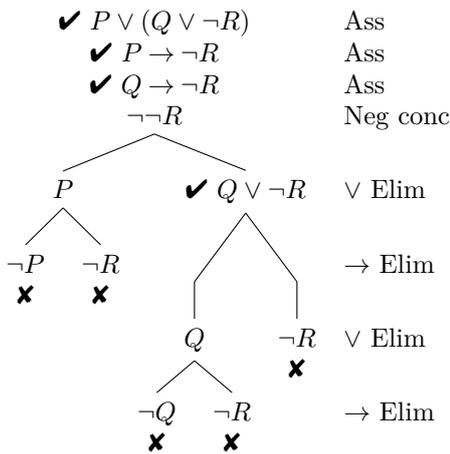
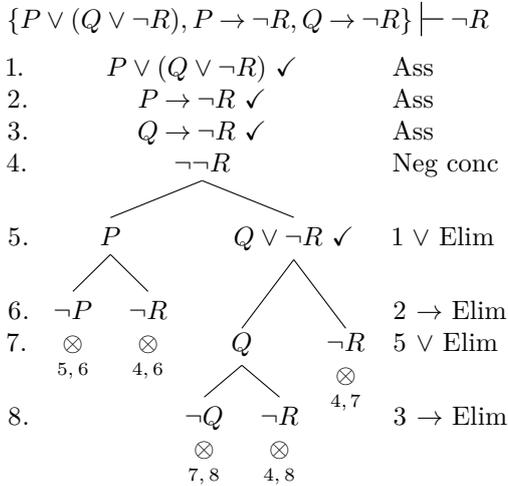
Furthermore, it is useful to have the option of *annotating* proof trees by numbering the lines of the proof on the left and entering the justification for each line on the right.

forest is a powerful and flexible package capable of all this and, indeed, a good deal more. It is not enormously difficult to customise particular trees to meet most of our desiderata. However, it is difficult to get things perfectly aligned even in simple cases, requires the insertion of 'phantom' nodes and management of several sub-trees in parallel (one for line numbers, one for the proof and one for the justifications). The process requires a good deal of manual intervention, trial-and-error and hard-coding of things it would be better to have L^AT_EX 2_ε manage for us, such as keeping count of lines and line references.

prooftrees aims to make it as easy to specify proof trees as it was to specify our initial tree using forest's default settings. The package supports a small number of options which can be configured to customise the output. The code for a prooftrees proof tree is shown in box 2, together with the output obtained using the default settings.

More extensive configuration can be achieved by utilising forest (Živanović 2016) and/or TikZ (Tantau 2015) directly. A sample of supported proof tree styles are shown in box 3. The package is *not* intended for the typesetting of proof trees which differ significantly in structure.

3 prooftrees: sample output



Either Alice saw nobody or she didn't see nobody.

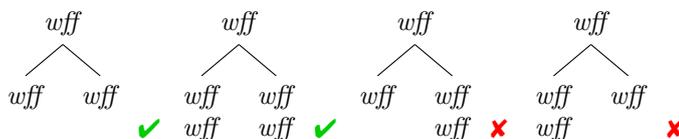
Alice saw nobody. \ Jones $\vee E$
 Alice didn't see Jones. $\vee E$

Alice didn't see nobody. $\vee E$
 Alice saw somebody. \checkmark Jones $\neg\neg E$
 Alice saw Jones. $\exists E$

2 Assumptions & Limitations

`prooftrees` makes certain assumptions about the nature of the proof system, \mathcal{L} , on which proofs are based.

- All derivation rules yield equal numbers of *wff*s on all branches.



If \mathcal{L} fails to satisfy this condition, `prooftrees` is likely to violate the requirements of affected derivation rules by splitting branches ‘mid-inference’.

- No derivation rule yields *wff*s on more than two branches.
- All derivation rules proceed in a downwards direction at an angle of -90° i.e. from north to south.
- Any justifications are set on the far right of the proof tree.
- Any line numbers are set on the far left of the proof tree.
- Justifications can refer only to earlier lines in the proof. `prooftrees` can typeset proofs if \mathcal{L} violates this condition, but the cross-referencing system explained in section 7.2 cannot be used for affected justifications.

`prooftrees` does not support the automatic breaking of proof trees across pages. Proof trees can be manually broken by using `line no shift` with an appropriate value for parts after the first (section 7.1). However, horizontal alignment across page breaks will not be consistent in this case.

In addition, `prooftrees` almost certainly relies on additional assumptions not articulated above and certainly depends on a feature of `forest` which its author classifies as experimental (`do dynamics`).

3 Typesetting a Proof Tree

After loading `prooftrees` in the document preamble:

```
% in document's preamble
\usepackage{prooftrees}
```

the `prooftree` environment is available for typesetting proof trees. This takes an argument used to specify a *tree preamble*, with the body of the environment consisting of a *tree specification* in `forest`’s notation. The *tree preamble* can be as simple as an empty argument — `{}` — or much more complex.

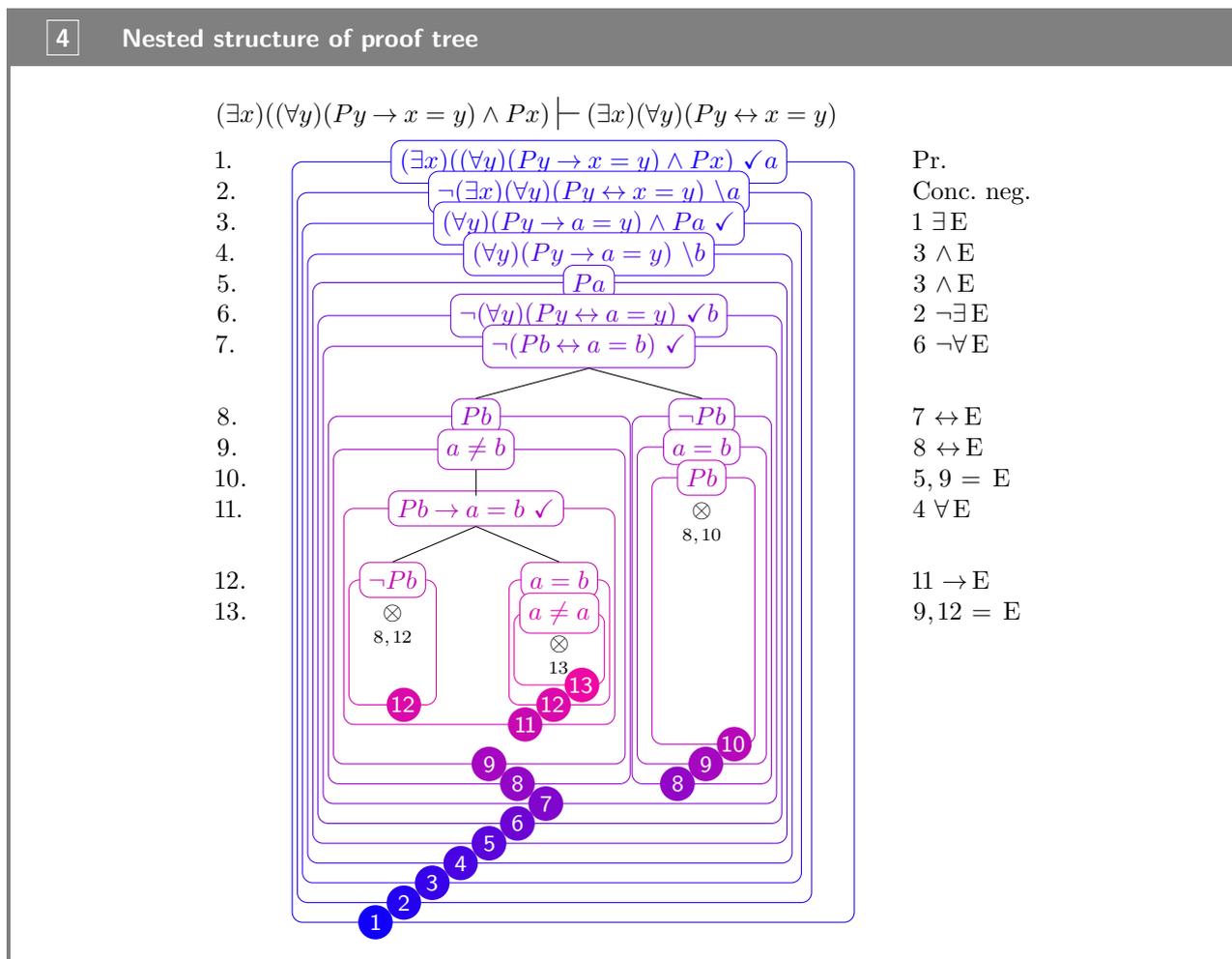
Customisation options and further details concerning loading and invocation are explained in section 4, section 5, section 6, section 7 and section 8. In this section, we begin by looking at a simple example using the default settings.

Suppose that we wish to typeset the proof tree for

$$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \vdash (\exists x)(\forall y)(Py \leftrightarrow x = y)$$

and we would like to typeset the entailment established by our proof at the top of the tree. Then we should begin like this:

```
\begin{prooftree}
{
  to prove={(\exists x)((\forallall y)(Py \liff x = y) \land Px) \sststile{}{} (\exists x)(\forallall y)(
Py \liff x = y)}
}
\end{prooftree}
```



That is all the preamble we want, so we move onto consider the *(tree specification)*. forest uses square brackets to specify trees' structures. To typeset a proof, think of it as consisting of nested trees, trunks upwards, and work from the outside in and the trunks down (box 4).

Starting with the outermost tree ① and the topmost trunk, we replace the with square brackets and enter the first *wff* inside, adding `just=Pr.` for the justification on the right and `checked=a` so that the line will be marked as discharged with *a* substituted for *x*. We also use forest's `name` to label the line for ease of reference later. (Technically, it is the node rather than the line which is named, but, for our purposes, this doesn't matter. forest will create a name if we don't specify one, but it will not necessarily be one we would have chosen for ease of use!)

```

\begin{prooftree}
{
  to prove={(\exists x)((\forallall y)(Py \lif x = y) \land Px) \sststile{}{} (\exists x)(\forallall y)(
Py \liff x = y)}
}
[{\exists x)((\forallall y)(Py \lif x = y) \land Px)}, checked=a, just=Pr., name=pr
]
\end{prooftree}
    
```

We can refer to this line later as `pr`.

We then consider the next tree ②. Its goes inside that for ①, so the square brackets containing the next *wff* go inside those we used for ①. Again, we add the justification with `just`, but we use `subs=a` rather than `checked=a` as we want to mark substitution of *a* for *x* without discharging the line. Again, we use `name` so

that we can refer to the line later as `neg conc`.

```
\begin{prooftree}
{
  to prove={{\exists x}({\forall y}(Py \lif x = y) \land Px) \sststile{}{} {\exists x}{\forall y}(
Py \liff x = y)}}
}
[{{\exists x}({\forall y}(Py \lif x = y) \land Px)}, checked=a, just=Pr., name=pr
 [{{\lnot (\exists x){\forall y}(Py \liff x = y)}, subs=a, just=Conc.\neg., name=neg conc
 ]
 ]
}\end{prooftree}
```

Turning to tree ③, we again note that its \square is nested within the previous two, so the square brackets for its *wff* need to be nested within those for the previous *wff*s. This time, we want to mark the line as discharged without substitution, so we simply use `checked` without a value. Since the justification for this line includes mathematics, we need to ensure that the relevant part of the justification is surrounded by `...$` or `\(...\)`. This justification also refers to an earlier line in the proof. We could write this as `just=1 $\exists\elim$`, but instead we use the name we assigned earlier with the referencing feature provided by `prooftrees`. To do this, we put the reference, `pr` after the rest of the justification, separating the two parts by a colon i.e. `$\exists\elim$:pr` and allow `prooftrees` to figure out the correct number.

```
\begin{prooftree}
{
  to prove={{\exists x}({\forall y}(Py \lif x = y) \land Px) \sststile{}{} {\exists x}{\forall y}(
Py \liff x = y)}}
}
[{{\exists x}({\forall y}(Py \lif x = y) \land Px)}, checked=a, just=Pr., name=pr
 [{{\lnot (\exists x){\forall y}(Py \liff x = y)}, subs=a, just=Conc.\neg., name=neg conc
 [{{\forall y}(Py \lif a = y) \land Pa}, checked, just=$\exists\elim$:pr
 ]
 ]
 ]
}\end{prooftree}
```

Continuing in the same way, we surround each of the *wff*s for ④, ⑤, ⑥ and ⑦ within square brackets nested within those surrounding the previous *wff* since each of the trees is nested within the previous one. Where necessary, we use `name` to label lines we wish to refer to later, but we also use `forest`'s *relative* naming system when this seems easier. For example, in the next line we add, we specify the justification as `just=$\land\elim$:!u`. `!u` tells `forest` that the reference specifies a relationship between the current line and the referenced one, rather than referring to the other line by name. `!u` refers to the current line's parent line — in this case, `{{\forall y}(Py \lif a = y) \land Pa}`, `checked, just=$\exists\elim$:pr`. `!uu` refers to the current line's parent line's parent line and so on.

```
\begin{prooftree}
{
  to prove={{\exists x}({\forall y}(Py \lif x = y) \land Px) \sststile{}{} {\exists x}{\forall y}(
Py \liff x = y)}}
}
[{{\exists x}({\forall y}(Py \lif x = y) \land Px)}, checked=a, just=Pr., name=pr
 [{{\lnot (\exists x){\forall y}(Py \liff x = y)}, subs=a, just=Conc.\neg., name=neg conc
 [{{\forall y}(Py \lif a = y) \land Pa}, checked, just=$\exists\elim$:pr
 [{{\forall y}(Py \lif a = y)}, subs=b, just=$\land\elim$:!u, name=mark
 [Pa, just=$\land\elim$:!uu, name=simple
 [{{\lnot (\forall y)(Py \liff a = y)}, checked=b, just=$\lnot\exists\elim$:neg conc
 [{{\lnot (Pb \liff a = b)}, checked, just=$\lnot\forall\elim$:!u
 ]
 ]
 ]
 ]
 ]
}\end{prooftree}
```

```

    ]
  ]
]
\end{prooftree}

```

Reaching ⑧, things get a little more complex since we now have not one, but *two* \square nested within ⑦. This means that we need *two* sets of square brackets for ⑧ — one for each of its two trees. Again, both of these should be nested within the square brackets for ⑦ but neither should be nested within the other because the trees for the two branches at ⑧ are distinct.

```

\begin{prooftree}
{
  to prove={(\exists x)((\forall y)(Py \liff x = y) \land Px) \sststile{}{} (\exists x)(\forall y)(
Py \liff x = y)}
}
[{\(\exists x)((\forall y)(Py \liff x = y) \land Px)}, checked=a, just=Pr., name=pr
[{\lnot (\exists x)(\forall y)(Py \liff x = y)}, subs=a, just=Conc.~neg., name=neg conc
[{\(\forall y)(Py \liff a = y) \land Pa}, checked, just=${\exists}\elim$:pr
[{\(\forall y)(Py \liff a = y)}, subs=b, just=${\land}\elim$:!u, name=mark
[Pa, just=${\land}\elim$:!uu, name=simple
[{\lnot (\forall y)(Py \liff a = y)}, checked=b, just=${\lnot}\exists\elim$:neg conc
[{\lnot (Pb \liff a = b)}, checked, just=${\lnot}\forall\elim$:!u
[Pb, just=${\liff}\elim$:!u, name=to Pb or not to Pb
]
[\lnot Pb
]
]
]
]
]
]
]
]
]
\end{prooftree}

```

At this point, we need to work separately or in parallel on each of our two branches since each constitutes its own tree. Turning to trees ⑨, each needs to be nested within the relevant tree ⑧, since each \square is nested within the applicable branch's tree. Hence, we nest square brackets for each of the *wffs* at ⑨ within the previous set.

```

\begin{prooftree}
{
  to prove={(\exists x)((\forall y)(Py \liff x = y) \land Px) \sststile{}{} (\exists x)(\forall y)(
Py \liff x = y)}
}
[{\(\exists x)((\forall y)(Py \liff x = y) \land Px)}, checked=a, just=Pr., name=pr
[{\lnot (\exists x)(\forall y)(Py \liff x = y)}, subs=a, just=Conc.~neg., name=neg conc
[{\(\forall y)(Py \liff a = y) \land Pa}, checked, just=${\exists}\elim$:pr
[{\(\forall y)(Py \liff a = y)}, subs=b, just=${\land}\elim$:!u, name=mark
[Pa, just=${\land}\elim$:!uu, name=simple
[{\lnot (\forall y)(Py \liff a = y)}, checked=b, just=${\lnot}\exists\elim$:neg conc
[{\lnot (Pb \liff a = b)}, checked, just=${\lnot}\forall\elim$:!u
[Pb, just=${\liff}\elim$:!u, name=to Pb or not to Pb
[a \neq b, just=${\liff}\elim$:!u
]
]
[\lnot Pb
[{a = b}
]
]
]
]
]
]
]
\end{prooftree}

```


$$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \vdash (\exists x)(\forall y)(Py \leftrightarrow x = y)$$

1.	$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \checkmark a$	Pr.
2.	$\neg(\exists x)(\forall y)(Py \leftrightarrow x = y) \setminus a$	Conc. neg.
3.	$(\forall y)(Py \rightarrow a = y) \wedge Pa \checkmark$	1 $\exists E$
4.	$(\forall y)(Py \rightarrow a = y) \setminus b$	3 $\wedge E$
5.	Pa	3 $\wedge E$
6.	$\neg(\forall y)(Py \leftrightarrow a = y) \checkmark b$	2 $\neg\exists E$
7.	$\neg(Pb \leftrightarrow a = b) \checkmark$	6 $\neg\forall E$
$\begin{array}{c} \swarrow \quad \searrow \\ Pb \quad \neg Pb \\ a \neq b \quad a = b \\ Pb \rightarrow a = b \checkmark \quad Pb \\ \swarrow \quad \searrow \quad \otimes \\ \neg Pb \quad a = b \quad 8,10 \\ \otimes \quad a \neq a \\ 8,11 \quad \otimes \\ \quad \quad 12 \end{array}$		
8.		7 $\leftrightarrow E$
9.		8 $\leftrightarrow E$
10.		4 $\forall E$; 5, 9 = E
11.		10 $\rightarrow E$
12.		9, 11 = E

prooftrees warns us about this:

Package prooftrees Warning: Merging conflicting justifications for line 10! Please examine the output carefully and use "move by" to move lines later in the proof if required. Details of how to do this are included in the documentation.

We would like line 10 in the left-hand branch to be moved down by one line, so we add `move by=1` to the relevant line of our proof. That is, we replace the line

$$[Pb \ \text{lif} \ a = b], \text{ checked, just=4 } \$\forall\text{forall}\text{elim}\$$$

by

$$[Pb \ \text{lif} \ a = b], \text{ checked, just}=\forall\text{forall}\text{elim}\$:\text{mark}, \text{ move by=1}$$

giving us the following code:

```
\begin{prooftree}
{
  to prove={{\exists x}({\forall y}(Py \text{ lif} x = y) \ \& Px) \sststile{}{} {\exists x}({\forall y}(
Py \text{ lif} x = y))}
}
[{{\exists x}({\forall y}(Py \text{ lif} x = y) \ \& Px)}, checked=a, just=Pr., name=pr
[{{\not} {\exists x}({\forall y}(Py \text{ lif} x = y))}, subs=a, just=Conc.\neg., name=neg conc
[{{\forall y}(Py \text{ lif} a = y) \ \& Pa}, checked, just=${\exists}\text{elim}\$:pr
[{{\forall y}(Py \text{ lif} a = y)}, subs=b, just=${\&}\text{elim}\$:!u, name=mark
[Pa, just=${\&}\text{elim}\$:!uu, name=simple
[{{\not} {\forall y}(Py \text{ lif} a = y)}, checked=b, just=${\not}\text{exists}\text{elim}\$:neg conc
[{{\not} (Pb \text{ lif} a = b)}, checked, just=${\not}\text{forall}\text{elim}\$:!u
[Pb, just=${\text{lif}}\text{elim}\$:!u, name=to Pb or not to Pb
[a \neq b, just=${\text{lif}}\text{elim}\$:!u
[ [Pb \text{ lif} a = b], checked, just=${\forall}\text{forall}\text{elim}\$:mark, move by=1
[ \not Pb, close={:to Pb or not to Pb,!c}, just=${\text{lif}}\text{elim}\$:!u
]
[ [a = b]
[ [a \neq a, close={:!c}, just=${\text{}}\text{elim}\$:!uuu,!u}
]
]
]
]
]
```

```

[\lnot Pb
  [{a = b}
    [Pb, just={\$=\elim$:{simple,!u}}, close={:to Pb or not to Pb,!c}
      ]
    ]
  ]
]
]
]
]
]
]
]
]
]
]
\end{prooftree}

```

which produces our desired result:

$$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \vdash (\exists x)(\forall y)(Py \leftrightarrow x = y)$$

1.	$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \checkmark a$	Pr.
2.	$\neg(\exists x)(\forall y)(Py \leftrightarrow x = y) \setminus a$	Conc. neg.
3.	$(\forall y)(Py \rightarrow a = y) \wedge Pa \checkmark$	1 $\exists E$
4.	$(\forall y)(Py \rightarrow a = y) \setminus b$	3 $\wedge E$
5.	Pa	3 $\wedge E$
6.	$\neg(\forall y)(Py \leftrightarrow a = y) \checkmark b$	2 $\neg\exists E$
7.	$\neg(Pb \leftrightarrow a = b) \checkmark$	6 $\neg\forall E$
$\begin{array}{cc} \swarrow & \searrow \\ Pb & \neg Pb \\ a \neq b & a = b \end{array}$		
8.		7 $\leftrightarrow E$
9.		8 $\leftrightarrow E$
10.		5, 9 = E
11.	$\begin{array}{cc} Pb \rightarrow a = b \checkmark & \otimes \\ & 8, 10 \end{array}$	4 $\forall E$
12.	$\begin{array}{cc} \neg Pb & a = b \\ \otimes & a \neq a \\ 8, 12 & \otimes \\ & 13 \end{array}$	11 $\rightarrow E$
13.		9, 12 = E

4 Loading the Package

To load the package simply add the following to your document's preamble.

```
\usepackage{prooftrees}
```

The package will load `forest` automatically. No options are currently supported but any given will be passed to `forest`.

Example: `\usepackage[debug]prooftrees`

would enable `forest`'s debugging.

If one or more of `forest`'s libraries are to be loaded, it is recommended that these be loaded separately and their defaults applied, if applicable, within a local `TEX` group so that they do not interfere with `prooftree`'s environment.

5 Invocation

`prooftree`
environment

```
\begin{prooftree}⟨tree preamble⟩⟨tree specification⟩\end{prooftree}
```

The `⟨tree preamble⟩` is used to specify any non-default options which should be applied to the tree. It may contain any code valid in the preamble of a regular `forest` tree, in addition to setting `prooftree` options. The preamble may be empty, but the argument is *required*¹. The `⟨tree specification⟩` specifies the tree in the bracket notation parsed by `forest`.

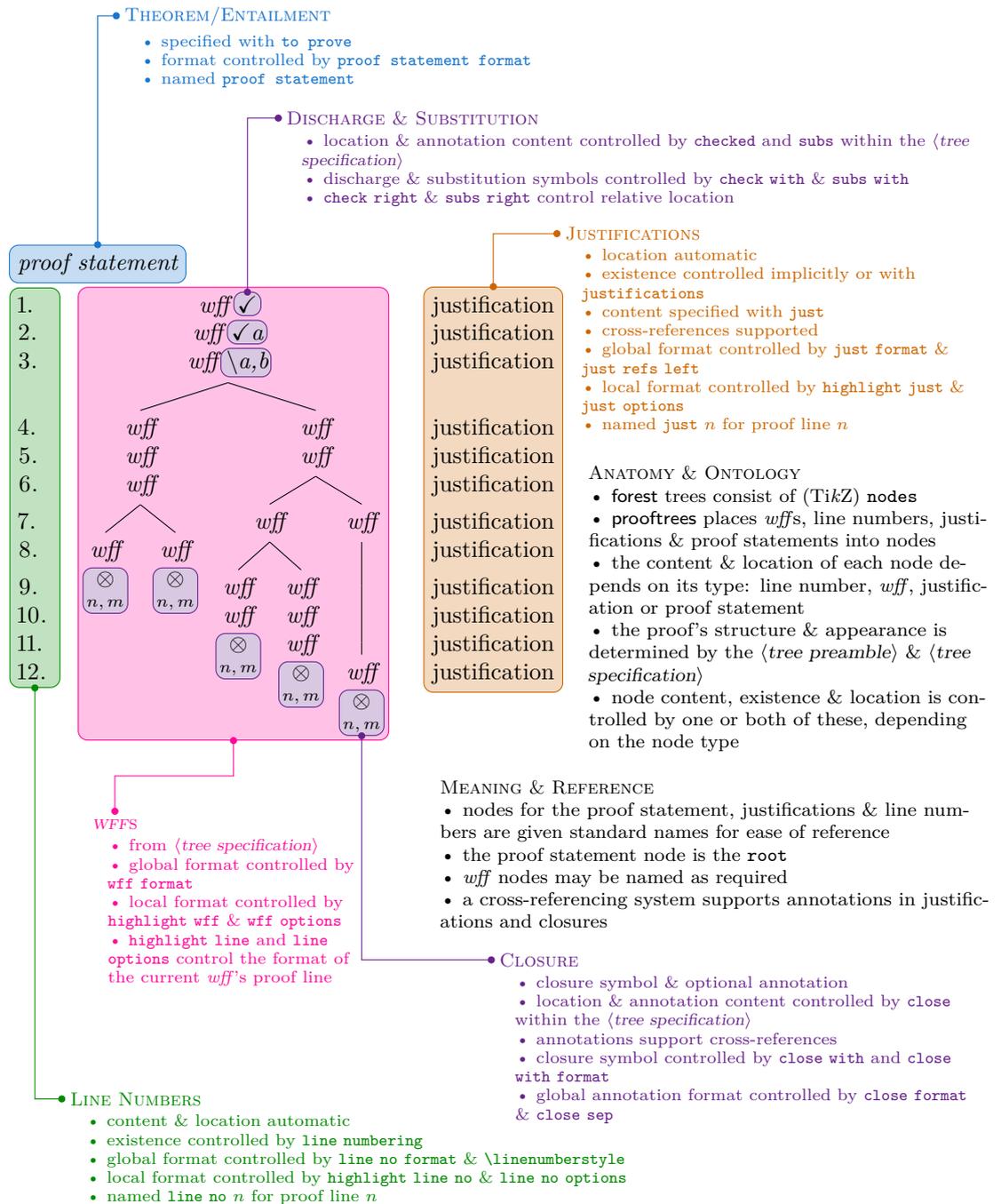
Users of `forest` should note that the environments `prooftree` and `forest` differ in important ways.

- *`prooftree`'s argument is mandatory.*
- *The tree's preamble cannot be given in the body of the environment.*
- *`\end{prooftree}` must follow the `⟨tree specification⟩` immediately.*

6 Proof Tree Anatomy

The following diagram provides an overview of the configuration and anatomy of a `prooftrees` proof tree. Detailed documentation is provided in section 7 and section 8.

¹Failure to specify a required argument does not always yield a compilation error in the case of environments. However, failure to specify required arguments to environments often fails to achieve the best consequences, even when it does not result in compilation failures, and will, therefore, be avoided by the prudent.



7 Options

Most configuration uses the standard key/value interface provided by TiKZ and extended by forest. These are divided into those which determine the overall appearance of the proof as a whole and those with more local effects.

7.1 Global Options

The following options affect the global style of the tree and should typically be set in the tree's preamble if non-default values are desired. The default values for the document can be set outside

the `prooftree` environment using `\forestset{⟨settings⟩}`. If *only* proof trees will be typeset, a default style can be configured using forest's default preamble.

`line numbering` = `true|false`
`not line numbering`
Forest boolean register

Default: `true`

This determines whether lines should be numbered. The default is to number lines. The following are equivalent to the default setting:

```
line numbering
line numbering=true
```

Either of the following will turn line numbering off:

```
not line numbering
line numbering=false
```

`justifications` = `true|false`
`not justifications`
Forest boolean register

This determines whether justifications for lines of the proof should be typeset to the right of the tree. It is rarely necessary to set this option explicitly as it will be automatically enabled if required. The only exception concerns a proof for which a line should be moved but no justifications are specified. In this case either of the following should be used to activate the option:

```
justifications
justifications=true
```

This is not necessary if `just` is used for any line of the proof.

`single branches` = `true|false`
`not single branches`
Forest boolean register

Default: `false`

This determines whether inference steps which do not result in at least two branches should draw and explicit branch. The default is to not draw single branches explicitly. The following are equivalent to the default setting:

```
not single branches
single branches=false
```

Either of the following will turn line numbering off:

```
single branches
single branches=true
```

`line no width` = $\langle dimension \rangle$
Forest dimension register

The maximum width of line numbers. By default, this is set to the width of the formatted line number 99.

Example: `line no width=20pt`

`just sep` = $\langle dimension \rangle$
Forest dimension register

Default: `1.5em`

Amount by which to shift justifications away from the tree. A larger value will shift the justifications further to the right, increasing their distance from the tree, while a smaller one will decrease this distance. Note that a negative value ought never be given. Although this will not cause an error, it may result in strange things happening. If you wish to decrease the distance

between the tree and the justifications further, please set `just sep` to zero and use the options provided by `forest` and/or `TikZ` to make further negative adjustments.

Example: `just sep=.5em`

`line no sep` = $\langle dimension \rangle$
Forest dimension register

Default: `1.5em`

Amount by which to shift line numbers away from the tree. A larger value will shift the line numbers further to the left, increasing their distance from the tree, while a smaller one will decrease this distance. Note that a negative value ought never be given. Although this will not cause an error, it may result in strange things happening. If you wish to decrease the distance between the tree and the line numbers further, please set `line no sep` to zero and use the options provided by `forest` and/or `TikZ` to make further negative adjustments.

Example: `line no sep=5pt`

`close sep` = $\langle dimension \rangle$
Forest dimension register

Default: `.75\baselineskip`

Distance between the symbol marking branch closure and any following annotation. If the format of such annotations is changed with `close format`, this dimension may require adjustment.

Example: `close sep=\baselineskip`

`line no shift` = $\langle integer \rangle$
Forest count register

Default: `0`

This value increments or decrements the number used for the first line of the proof. By default, line numbering starts at 1.

Example: `line no shift=3`

would begin numbering the lines at 4.

`zero start` Start line numbering from 0 rather than 1. The following are equivalent:
Forest style

```
zero start
line no shift=-1
```

`to prove` = $\langle wff \rangle$
Forest style

Statement of theorem or entailment to be typeset above the proof. In many cases, it will be necessary to enclose the statement in curly brackets.

Example: `to prove={\sststyle{}} P \lif P`

By default, the content is expected to be suitable for typesetting in maths mode and should *not*, therefore, be enclosed by dollar signs or equivalent.

`check with` = $\langle symbol \rangle$
Forest toks register

Default: `\ensuremath{\checkmark}` (✓)

Symbol with which to mark discharged lines.

Example: `check with={\text{\ding{52}}}`

Within the tree, `checked` is used to identify discharged lines.

`check right` = `true|false`
`not check right`
Forest boolean register

Default: `true`

Determines whether the symbol indicating that a line is discharged should be placed to the right of the *wff*. The alternative is, unsurprisingly, to place it to the left of the *wff*. The following are equivalent to the default setting:

```
check right
check right=true
```

check left Set `check right=false`. The following are equivalent ways to place the markers to the left:
Forest style

```
check right=false
not check right
check left
```

close with = $\langle symbol \rangle$
Forest toks register

Default: `\ensuremath{\otimes}` (\otimes)

Symbol with which to close branches.

Example: `close with={\ensuremath{\ast}}`

Within the tree, `close` is used to identify closed branches.

close with format = $\langle key-value list \rangle$
Forest keylist register

Additional TikZ keys to apply to the closure symbol. Empty by default.

Example: `close with format={red, font=}`

To replace a previously set value, rather than adding to it, use `close with format'` rather than `close with format`.

close format = $\langle key-value list \rangle$
Forest keylist register

Default: `font=\scriptsize`

Additional TikZ keys to apply to any annotation following closure of a branch.

Example: `close format={font=\footnotesize\sffamily, text=gray!75}`

To replace the default value of `close format`, rather than adding to it, use `close format'` rather than `close format`.

Example: `close format'={text=red}`

will produce red annotations in the default font size, whereas

Example: `close format={text=red}`

will produce red annotations in `\scriptsize`.

subs with = $\langle symbol \rangle$
Forest toks register

Default: `\ensuremath{\backslash}` (\backslash)

Symbol to indicate variable substitution.

Example: `\text{:}`

Within the tree, `subs` is used to indicate variable substitution.

subs right = `true|false`
not subs right
Forest boolean register

Default: `true`

Determines whether variable substitution should be indicated to the right of the *wff*. The alternative is, again, to place it to the left of the *wff*. The following are equivalent to the default setting:

```
subs right
subs right=true
```

subs left Set `subs right=false`. The following are equivalent ways to place the annotations to the left:
Forest style

```
subs right=false
not subs right
subs left
```

just refs left = true|false
not just refs left
Forest boolean register

Default: true

Determines whether line number references should be placed to the left of justifications. The alternative is to place them to the right of justifications. The following are equivalent to the default setting:

```
just refs left
just refs left=true
```

just refs right Set `just refs left=false`. The following are equivalent ways to place the references to the right:
Forest style

```
just refs left=false
not just refs left
just refs right
```

Note that this setting *only affects the placement of line numbers specified using the cross-referencing system* explained in section 7.2. Hard-coded line numbers in justifications will be typeset as is.

just format = ⟨key-value list⟩
Forest keylist register

Additional TikZ keys to apply to line justifications. Empty by default.

Example: `just format={red, font=}`

To replace a previously set value, rather than adding to it, use `just format'` rather than `just format`.

line no format = ⟨key-value list⟩
Forest keylist register

Additional TikZ keys to apply to line numbers. Empty by default.

Example: `line no format={align=right, text=gray}`

To replace a previously set value, rather than adding to it, use `line no format'` rather than `line no format`. To change the way the number itself is formatted — to eliminate the dot, for example, or to put the number in brackets — redefine `\linenumberstyle` (see section 8).

wff format = ⟨key-value list⟩
Forest keylist register

Additional TikZ keys to apply to *wffs*. Empty by default.

Example: `wff format={draw=orange}`

To replace a previously set value, rather than adding to it, use `wff format'` rather than `wff format`.

proof statement format = ⟨key-value list⟩
Forest keylist register

Additional TikZ keys to apply to the proof statement. Empty by default.

Example: `proof statement format={text=gray, draw=gray}`

To replace a previously set value, rather than adding to it, use `proof statement format'` rather than `proof statement format`.

`highlight format`
Forest autowrapped toks register

= \langle key-value list \rangle

Default: `draw=gray, rounded corners`

Additional TikZ keys to apply to highlighted *wff*s.

Example: `highlight format={text=red}`

To apply highlighting, use the `highlight wff`, `highlight just`, `highlight line no` and/or `highlight line` keys (see section 7.2).

`merge delimiter`
Forest toks register

= \langle punctuation \rangle

Default: `\text{; } (;)`

Punctuation to separate distinct justifications for a single proof line. Note that `prooftrees` will issue a warning if it detects different justifications for a single proof line and will suggest using `move by` to avoid the need for merging justifications. In general, justifications ought not be merged because it is then less clear to which *wff*(s) each justification applies. Moreover, later references to the proof line will be similarly ambiguous. That is, `merge delimiter` ought almost never be necessary because it is almost always better to restructure the proof to avoid ambiguity.

7.2 Local Options

The following options affect the local structure or appearance of the tree and should typically be passed as options to the relevant node(s) within the tree.

`grouped`
`not grouped`
Forest boolean option

Indicate that a line is not an inference. When `single branches` is false, as it is with the default settings, this key is applied automatically and need not be given in the specification of the tree. When `single branches` is true, however, this key must be specified for any line which ought not be treated as an inference.

Example: `grouped`

`checked`
Forest style

Mark a complex *wff* as resolved, discharging the line.

Example: `checked`

`checked`
Forest style

= \langle name \rangle

Existential elimination, discharge by substituting \langle name \rangle .

Example: `checked=a`

`close`
Forest style

Close branch.

Example: `close`

`close`
Forest style

= \langle annotation \rangle

= \langle annotation prefix \rangle : \langle references \rangle

Close branch with annotation. In the simplest case, \langle annotation \rangle contains no colon and is typeset simply as it is. Any required references to other lines of the proof are assumed to be given explicitly.

Example: `close={12,14}`

If \langle annotation \rangle includes a colon, `prooftrees` assumes that it is of the form \langle annotation prefix \rangle : \langle references \rangle . In this case, the material prior to the colon should include material to be typeset before the line numbers and the material following the colon should consist of one or more references to other lines in the proof. In typical cases, no prefix will be required so that the

colon will be the first character. In case there is a prefix, `prooftrees` will insert a space prior to the line numbers. $\langle references \rangle$ may consist of either forest names (e.g. given by `name= $\langle name label \rangle$`) and then used as $\langle name label \rangle$) or forest relative node names (e.g. $\langle nodewalk \rangle$) or a mixture.

Example: `close={:negated conclusion}`

where `name=negated conclusion` was used to label an earlier proof line `negated conclusion`. If multiple references are given, they should be separated by commas and either $\langle references \rangle$ or the entire $\langle annotation \rangle$ must be enclosed in curly brackets, as is usual for `TikZ` and forest values containing commas.

Example: `close={!:c,!uuu}`

`subs`
Forest style = $\langle name \rangle / \langle names \rangle$

Universal instantiation, instantiate with $\langle name \rangle$ or $\langle names \rangle$.

Example: `subs={a,b}`

`just`
Forest autowrapped toks option = $\langle justification \rangle$
= $\langle justification prefix/suffix \rangle : \langle references \rangle$

Justification for inference. This is typeset in text mode. Hence, mathematical expressions must be enclosed suitably in dollar signs or equivalent. In the simplest case, $\langle justification \rangle$ contains no colon and is typeset simply as it is. Any required references to other lines of the proof are assumed to be given explicitly.

Example: `just=3 \lorD`

If $\langle justification \rangle$ includes a colon, `prooftrees` assumes that it is of the form $\langle justification prefix/suffix \rangle : \langle references \rangle$. In this case, the material prior to the colon should include material to be typeset before or after the line numbers and the material following the colon should consist of one or more references to other lines in the proof. Whether the material prior to the colon is interpreted as a $\langle justification prefix \rangle$ or a $\langle justification suffix \rangle$ depends on the value of `just refs left`. $\langle references \rangle$ may consist of either forest names (e.g. given by `name= $\langle name label \rangle$`) and then used as $\langle name label \rangle$) or forest relative node names (e.g. $\langle nodewalk \rangle$) or a mixture. If multiple references are given, they should be separated by commas and $\langle references \rangle$ must be enclosed in curly brackets. If `just refs left` is true, as it is by default, then the appropriate line number(s) will be typeset before the $\langle justification suffix \rangle$.

Example: `just=$\lnot\exists\elim:{!uu,!u}`

If `just refs left` is false, then the appropriate line number(s) will be typeset after the $\langle justification prefix \rangle$.

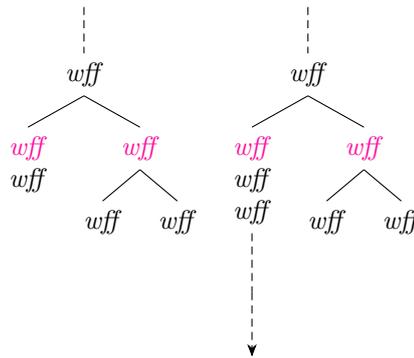
Example: `just=From:bertha`

`move by`
Forest style = $\langle positive integer \rangle$

Move the content of the current line $\langle positive integer \rangle$ lines later in the proof. If the current line has a justification and the content is moved, the justification will be moved with the line. Later lines in the same branch will be moved appropriately, along with their justifications.

Example: `move by=3`

Note that, in many cases, `prooftrees` will automatically move lines later in the proof. It does this when it detects a condition in which it expects conflicting justifications may be required for a line while initially parsing the tree. Essentially, `prooftrees` tries to detect cases in which a branch is followed closely by asymmetry in the structure of the branches. This happens, for example, when the first branch's first *wff* is followed by a single *wff*, while the second branch's first *wff* is followed by another branch. Diagrammatically:



In this case, `prooftrees` tries to adjust the tree by moving lines appropriately if required.

However, this detection is merely structural — `prooftrees` does not examine the content of the `wff`s or justifications for this purpose. Nor does it look for slightly more distant structural asymmetries, conflicting justifications in the absence of structural asymmetry or potential conflicts with justifications for lines in other, more distant parallel branches. Although it is not that difficult to detect the *need* to move lines in a greater proportion of cases, the problem lies in providing general rules for deciding *how* to resolve such conflicts. (Indeed, some such conflicts might be better left unresolved e.g. to fit a proof on a single Beamer slide.) In these cases, a human must tell `prooftrees` if something should be moved, what should be moved and how far it should be moved.

Because simple cases are automatically detected, it is best to typeset the proof before deciding whether or where to use this option since `prooftrees` will assume that this option specifies movements which are required *in addition to* those it automatically detects. Attempting to move a line ‘too far’ is not advisable. `prooftrees` tries to simply ignore such instructions, but the results are likely to be unpredictable.

Not moving a line far enough — or failing to move a line at all — may result in the content of one justification being combined with that of another. This happens if `just` is specified more than once for the same proof line with differing content. `prooftrees` *does* examine the content of justifications for *this* purpose. When conflicting justifications are detected for the same proof line, the justifications are merged and a warning issued suggesting the use of `move by`.

<code>highlight wff</code>	Highlight <code>wff</code> .
<code>not highlight wff</code>	Example: <code>highlight wff</code>
<i>Forest boolean option</i>	
<code>highlight just</code>	Highlight justification.
<code>not highlight just</code>	Example: <code>highlight just</code>
<i>Forest boolean option</i>	
<code>highlight line no</code>	Highlight line number.
<code>not highlight line no</code>	Example: <code>highlight line no</code>
<i>Forest boolean option</i>	
<code>highlight line</code>	Highlight proof line.
<code>not highlight line</code>	Example: <code>highlight line</code>
<i>Forest boolean option</i>	
<code>line no options</code>	= \langle key-value list \rangle
<i>Forest autowrapped toks option</i>	Additional TikZ keys to apply to the line number for this line.
	Example: <code>line no options={blue}</code>
<code>just options</code>	= \langle key-value list \rangle
<i>Forest autowrapped toks option</i>	Additional TikZ keys to apply to the justification for this line.

Example: `just options={draw, font=\bfseries}`

`wff options`

= *<key-value list>*

Forest autowrapped toks option

Additional TikZ keys to apply to the *wff* for this line.

Example: `wff options={magenta, draw}`

Note that this key is provided primarily for symmetry as it is faster to simply give the options directly to `forest` to pass on to TikZ. Unless `wff format` is set to a non-default value, the following are equivalent:

```
wff options={magenta, draw}
magenta, draw
```

`line options`

= *<key-value list>*

Forest autowrapped toks option

Additional TikZ keys to apply to this proof line.

Example: `line options={draw, rounded corners}`

8 Macros

`\linenumberstyle`

{*<number>*}

macro

This macro is responsible for formatting the line numbers. The default definition is

```
\newcommand*\linenumberstyle[1]{#1.}
```

It may be redefined with `\renewcommand*` in the usual way. For example, if for some reason you would like bold line numbers, try

```
\renewcommand*\linenumberstyle[1]{\textbf{#1.}}
```

9 Version History

0.5

Significant re-implementation leveraging the new argument processing facilities in `forest` 2.1. This significantly improves performance as the code is executed much faster than the previous `pgfmath` implementation.

0.41

Update for compatibility with `forest` 2.1.

0.4

Bug fix release:

- `line no shift` was broken;
- in some cases, an edge was drawn where no edge belonged.

0.3

First CTAN release.

References

Hodges, Wilfred (1977, 1991). *Logic: An Introduction to Elementary Logic*. Penguin.

Tantau, Till (2015). *The TikZ and PGF Packages. Manual for Version 3.0.1a*. 3.0.1a. 29th Aug. 2015. URL: <http://sourceforge.net/projects/pgf>.

Živanović, Sašo (2016). *Forest: A PGF/TikZ-Based Package for Drawing Linguistic Trees*. 2.0.2. 4th Mar. 2016. URL: <http://spj.ff.uni-lj.si/zivanovic/>.

Index

Features are sorted by kind. Page references are given for both definitions and comments on use.

- FOREST AUTOWRAPPED TOKS OPTIONS
 - just, [6](#), [9](#), [16](#), [21](#), [22](#)
 - just options, [15](#), [22](#)
 - line no options, [15](#), [22](#)
 - line options, [15](#), [23](#)
 - wff options, [15](#), [23](#)
- FOREST AUTOWRAPPED TOKS REGISTERS
 - highlight format, [20](#)
- FOREST BOOLEAN OPTIONS
 - grouped, [20](#)
 - highlight just, [15](#), [20](#), [22](#)
 - highlight line, [15](#), [20](#), [22](#)
 - highlight line no, [15](#), [20](#), [22](#)
 - highlight wff, [15](#), [20](#), [22](#)
 - not grouped, [20](#)
 - not highlight line, [22](#)
 - not highlight line no, [22](#)
 - not highlight just, [22](#)
 - not highlight wff, [22](#)
- FOREST BOOLEAN REGISTERS
 - check right, [15](#), [17](#), [18](#)
 - just refs left, [15](#), [19](#), [21](#)
 - justifications, [16](#)
 - line numbering, [16](#)
 - not check right, [17](#)
 - not just refs left, [19](#)
 - not justifications, [16](#)
 - not line numbering, [16](#)
 - not single branches, [16](#)
 - not subs right, [18](#)
 - single branches, [16](#), [20](#)
 - subs right, [15](#), [18](#), [19](#)
- FOREST COUNT REGISTERS
 - line no shift, [5](#), [17](#), [23](#)
- FOREST DIMENSION REGISTERS
 - close sep, [15](#), [17](#)
 - just sep, [16](#), [17](#)
 - line no sep, [17](#)
 - line no width, [16](#)
- FOREST KEYLIST REGISTERS
 - close format, [15](#), [17](#), [18](#)
 - close format', [18](#)
 - close with format, [15](#), [18](#)
 - close with format', [18](#)
 - just format, [15](#), [19](#)
 - just format', [19](#)
 - line no format, [15](#), [19](#)
 - line no format', [19](#)
 - proof statement format, [15](#), [19](#), [20](#)
 - proof statement format', [20](#)
 - wff format, [15](#), [19](#), [23](#)
 - wff format', [19](#)
- FOREST STYLES
 - check left, [18](#)
 - checked, [7](#), [15](#), [17](#), [20](#)
 - close, [15](#), [18](#), [20](#)
 - just refs right, [19](#)
 - move by, [20](#)–[22](#)
 - subs, [15](#), [18](#), [21](#)
 - subs left, [19](#)
 - to prove, [17](#)
 - zero start, [17](#)
- FOREST TOKS REGISTERS
 - check with, [15](#), [17](#)
 - close with, [15](#), [18](#)
 - merge delimiter, [20](#)
 - subs with, [15](#), [18](#)
- ENVIRONMENTS
 - prooftree, [14](#)
- MACROS
 - `\linenumberstyle`, [23](#)